Quantum
LEARNING ACADEMY

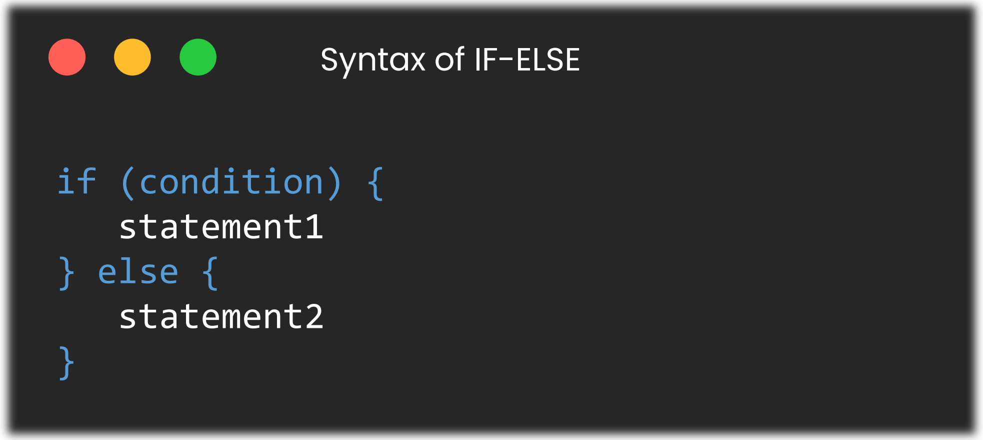
</>

CONDITIONAL STATEMENTS & LOOPS

IF, IF-ELSE, SWITCH STATEMENTS AND WHILE, FOR LOOPS IN JAVASCRIPT

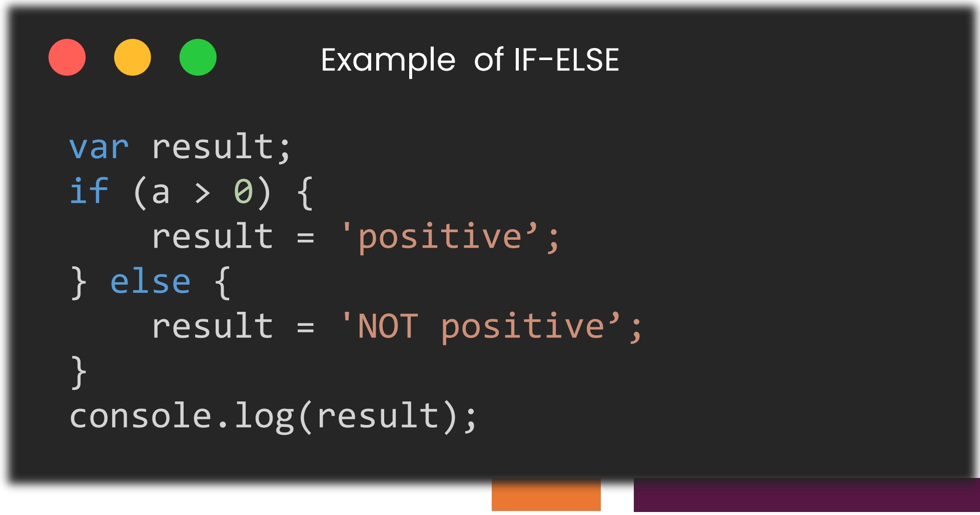
Conditional Statements

- Conditional statements are used to perform different actions based on different conditions. There are three types of conditional statements.
  1. If-else Statement
  2. Switch Statement
  3. Ternary Operator
- **IF-ELSE Statements:**
  - The if statement executes a statement if a specified condition is truthy. If the condition is falsy, the code in else block is executed.
  - All values are truthy except `false`, `0`, `-0`, `0n`, `""`, `null`, `undefined`, and `NaN`



Syntax of IF-ELSE

```
if (condition) {
   statement1
} else {
   statement2
}
```



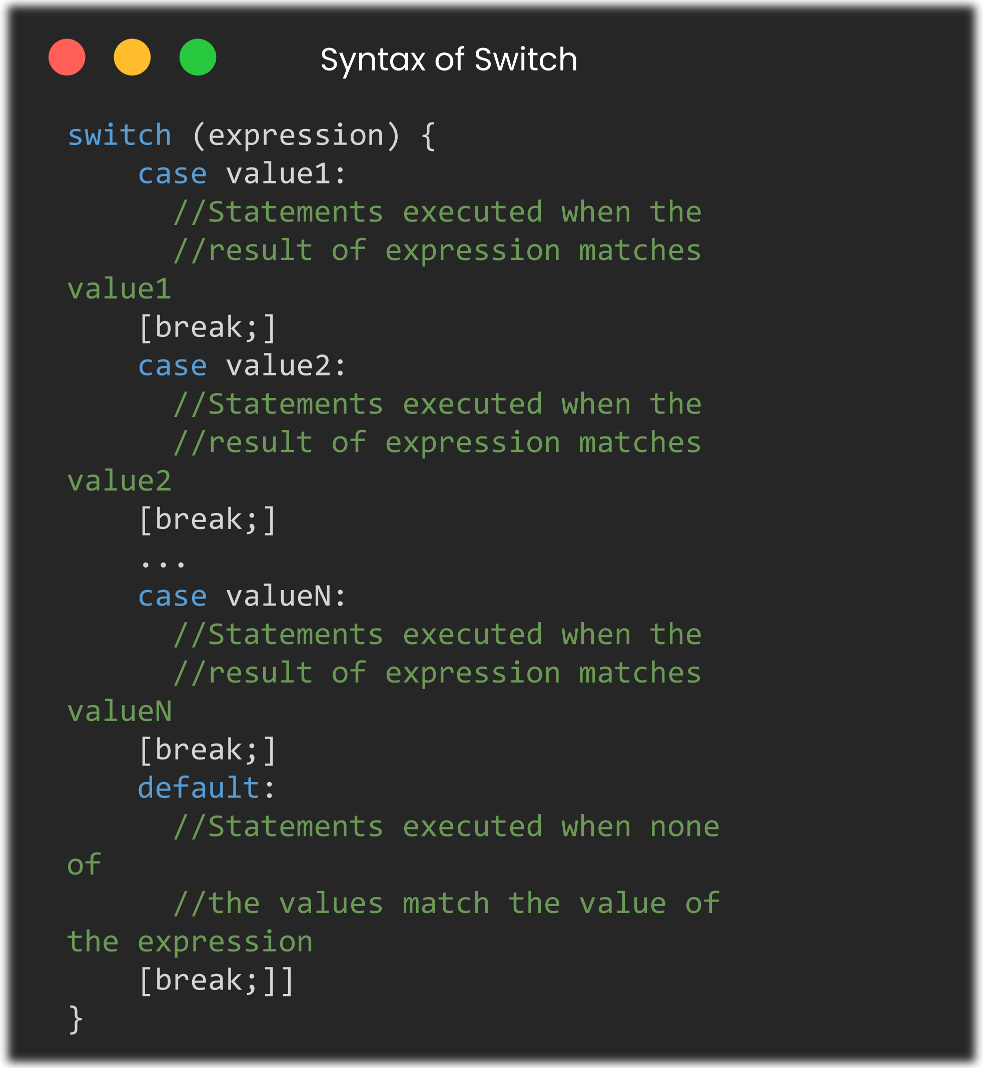
Example of IF-ELSE

```
var result;
if (a > 0) {
    result = 'positive’;
} else {
    result = 'NOT positive’;
}
console.log(result);
```

Switch

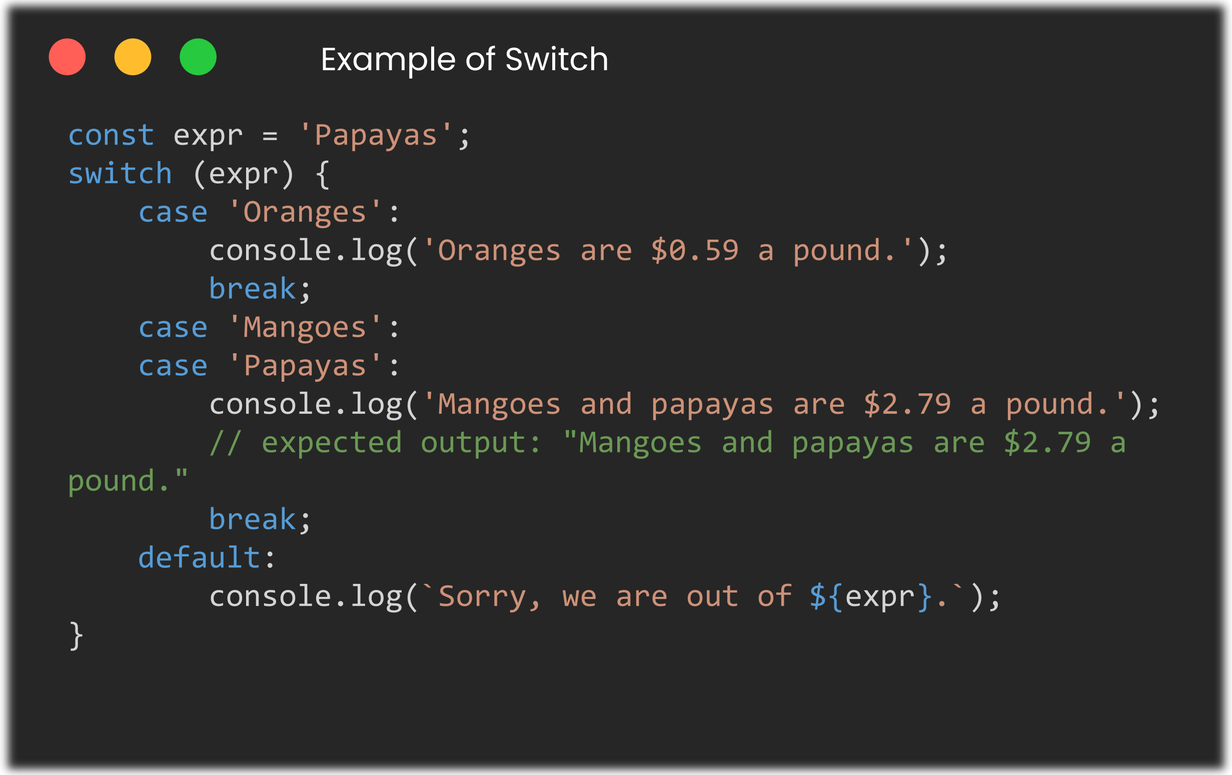
- **SWITCH Statements:**
  - The switch statement evaluates an expression, matching the expression's value to a case clause, and executes statements associated with that case, as well as statements in cases that follow the matching case.
  - A switch statement first evaluates its expression. It then looks for the first case clause whose expression evaluates to the same value as the result of the input expression (using the strict comparison, ===) and transfers control to that clause, executing the associated statements. (If multiple cases match the provided value, the first case that matches is selected, even if the cases are not equal to each other.).
  - If no matching case clause is found, the program looks for the optional default clause, and if found, transfers control to that clause, executing the associated statements. If no default clause is found, the program continues execution at the statement following the end of switch. By convention, the default clause is the last clause, but it does not need to be so.
  - The optional break statement associated with each case label ensures that the program breaks out of switch once the matched statement is executed and continues execution at the statement following switch. If break is omitted, the program continues execution at the next statement in the switch statement. The break statement is not required if a return statement precedes it.

Switch



Syntax of Switch

```
switch (expression) {
    case value1:
      //Statements executed when the
      //result of expression matches
value1
    [break;]
    case value2:
      //Statements executed when the
      //result of expression matches
value2
    [break;]
    ...
    case valueN:
      //Statements executed when the
      //result of expression matches
valueN
    [break;]
    default:
      //Statements executed when none
of
      //the values match the value of
the expression
    [break;]]
}
```

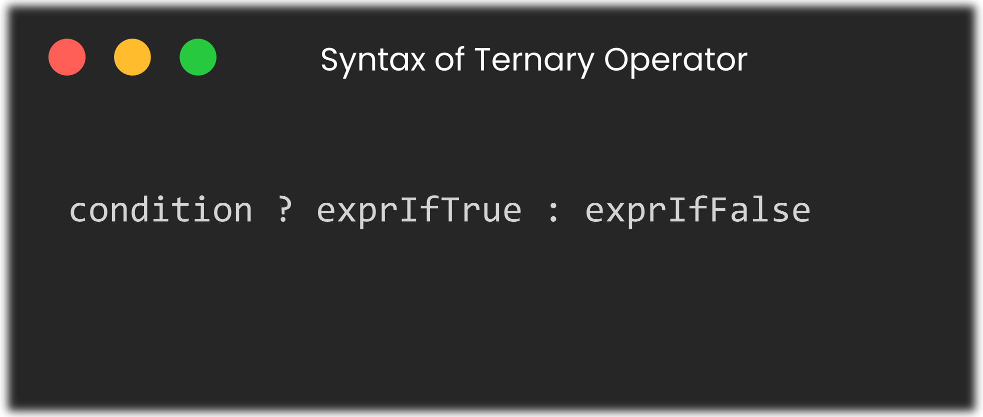


Example of Switch

```
const expr = 'Papayas';
switch (expr) {
    case 'Oranges':
        console.log('Oranges are $0.59 a pound.');
        break;
    case 'Mangoes':
    case 'Papayas':
        console.log('Mangoes and papayas are $2.79 a pound.');
        // expected output: "Mangoes and papayas are $2.79 a
pound."
        break;
    default:
        console.log(`Sorry, we are out of ${expr}.`);
}
```

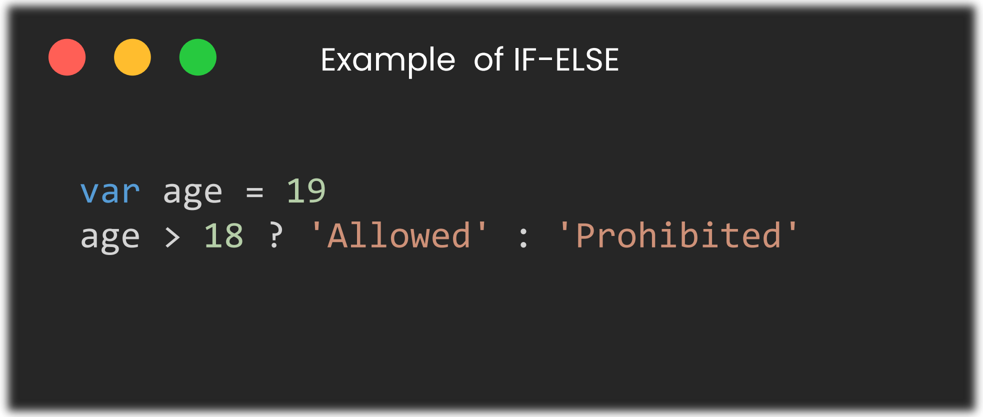
Ternary Operator

- The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark (?), then an expression to execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy. This operator is frequently used as an alternative to an if...else statement.
- Besides false, possible falsy expressions are: `null`, `NaN`, 0, the empty string (""), and `undefined`. If condition is any of these, the result of the conditional expression will be the result of executing the expression exprIfFalse.



Syntax of Ternary Operator

```
condition ? exprIfTrue : exprIfFalse
```



Example of IF-ELSE

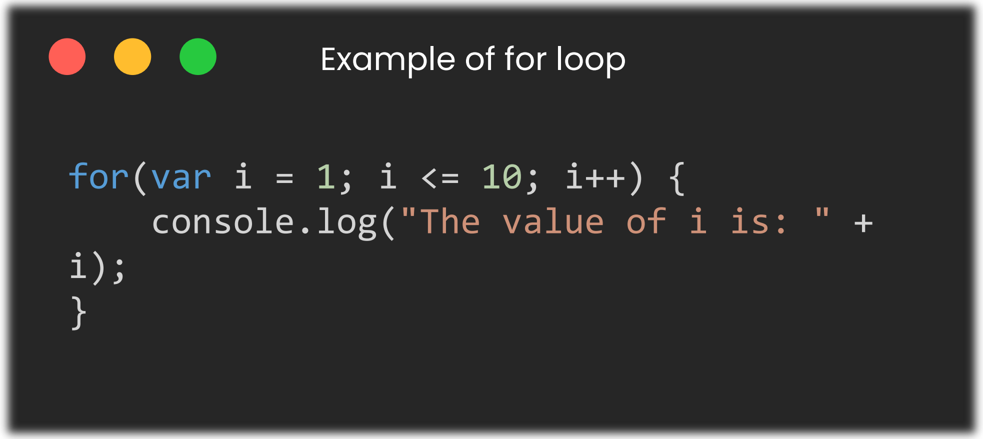
```
var age = 19
age > 18 ? 'Allowed' : 'Prohibited'
```

Loops

- Loops offer a quick and easy way to do something repeatedly.
- There are many different kinds of loops, but they all essentially do the same thing: they repeat an action some number of times. (Note that it's possible that number could be zero!)
- The various loop mechanisms offer different ways to determine the start and end points of the loop. There are various situations that are more easily served by one type of loop over the others.
- The statements for loops provided in JavaScript are:
  1. for loop
  2. While loop
  3. for…in loop
  4. for..of loop
  5. do…while loop

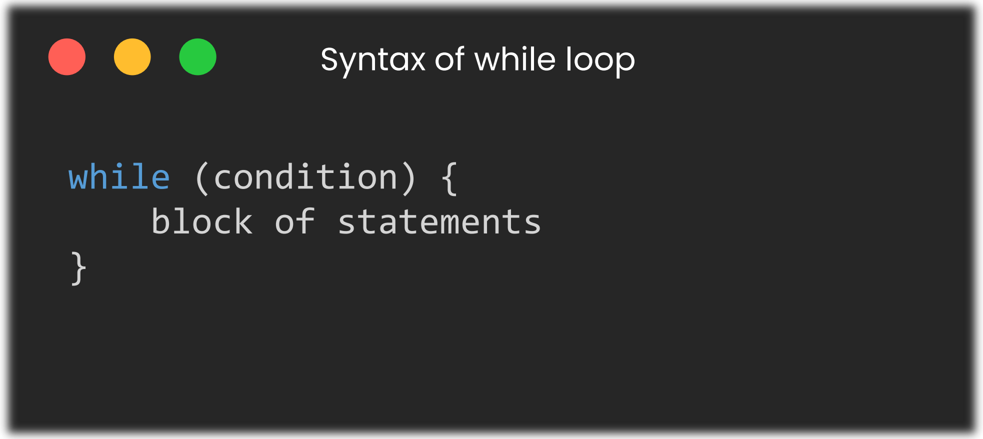
Loops

- **For loop:**
- A for loop repeats until a specified condition evaluates to false. The JavaScript for loop is similar to the Java and C for loop.



Example of for loop

```
for(var i = 1; i <= 10; i++) {
    console.log("The value of i is: " +
i);
}
```



Syntax of while loop

```
while (condition) {
    block of statements
}
```

- **While loop:**
- A while statement executes its statements as long as a specified condition evaluates to true. A while statement looks as follows:
- If the condition becomes false, statement within the loop stops executing and control passes to the statement following the loop.

Any Questions?

THANKS!

Instructor: Umair Syed
Assistant: Faisal Tahir
itsumairsyed@gmail.com
+92 309 0403499
Quantum Learning Academy